

grain

D Language Library for Deep Learning

ML Meetup KANSAI #3 LT

4. Oct. 2018

D Language for Deep Learning

great language

- ▶ **like C++**: fast, strongly typed, LLVM/GCC backend
- ▶ **like Python**: simple, lightweight, jupyter support¹

great libraries

- ▶ **mir**: N-D slice algorithm², zero-cost numpy conversion³
- ▶ **dcompute**: CUDA kernel DSL⁴

¹<https://github.com/kaleidicassociates/jupyterd>

²<https://github.com/libmir/mir-algorithm>

³<https://github.com/ShigekiKarita/mir-pybuffer>

⁴<https://github.com/libmir/dcompute>

grain

deep learning framework for D language

- ▶ <https://github.com/ShigekiKarita/grain>
- ▶ boost software license 1.0

philosophy

- ▶ **DYNAMIC**: like chainer and pytorch
- ▶ **SAFE**: statically typed variable and function
- ▶ **LIGHT**: simple like Python, small like C++
- ▶ **FAST**: mir and CUDA backend

grain is **dynamic**

like chainer ...

```
1 foreach (epoch; 0 .. 10) {  
2     foreach (i; niter.permutation) {  
3         auto xs = inputs[i].variable;  
4         auto ts = targets[i].variable;  
5         auto ys = model(xs);  
6         auto loss = crossEntropy(ys, ts);  
7         auto acc = accuracy(ys, ts);  
8         model.zeroGrad();  
9         loss.backward();  
10        optimizer.update();  
11    }  
12 }
```

grain is **dynamic**

like chainer **but statically typed and optimized.**

```
1 foreach (epoch; 0 .. 10) {  
2     foreach (i; niter.permutation) {  
3         Variable!(float, 3, HostStorage) xs = inputs[i].variable;  
4         Variable!(int, 1, HostStorage) ts = targets[i].variable;  
5         Variable!(float, 2, HostStorage) ys = model(xs);  
6         Variable!(float, 0, HostStorage) loss = crossEntropy(ys, ts);  
7         float acc = accuracy(ys, ts);  
8         model.zeroGrad();  
9         loss.backward();  
10        optimizer.update();  
11    }  
12 }
```

grain is **safe**

function is also statically typed and optimized.

```
1 struct Sigmoid(T, size_t dim) {  
2     Variable!(T, dim, HostStorage) y;  
3     // D compiler guarantees that nothing throws exception  
4     nothrow forward(Variable!(T, dim, HostStorage) x) {  
5         auto y = x.sliced.map!(a => tanh(a * 0.5) * 0.5 + 0.5)  
6             .slice.variable(x.requiresGrad);  
7         if (x.requiresGrad) this.y = y;  
8         return y;  
9     }  
10    nothrow backward(Variable!(T, dim, HostStorage) gy) {  
11        auto ys = this.y.sliced;  
12        return slice((1.0 - ys) * ys * gy.sliced).variable;  
13    }  
14    mixin FunctionCommon; // inject type checking  
15 }
```

grain is **safe**

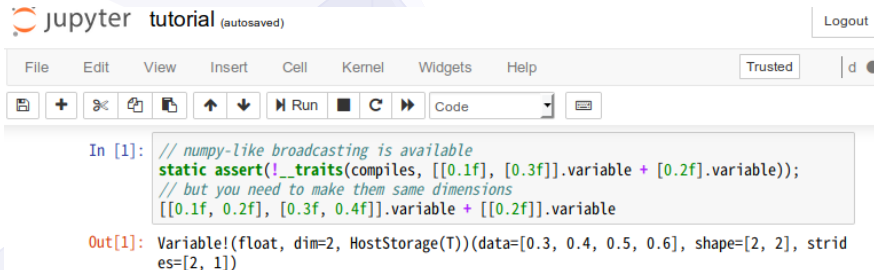
Chainer/PyTorch issues

- ▶ runtime overhead
 - ▶ for-loop, dynamic dispatch, func call
- ▶ runtime error:
 - ▶ type error, exception safety, memory leak

D solved these issues by compiler techniques and GC

grain is a lightweight language

Jupyter notebook support ⁵



The screenshot shows a Jupyter Notebook window titled "jupyter tutorial (autosaved)" with a "Logout" button in the top right. The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The toolbar contains icons for saving, adding, deleting, and running cells, along with a dropdown menu currently set to "Code".

The active code cell contains the following Python code:

```
In [1]: // numpy-like broadcasting is available  
static assert(!_traits(compiles, [[0.1f], [0.3f]].variable + [0.2f].variable));  
// but you need to make them same dimensions  
[[0.1f, 0.2f], [0.3f, 0.4f]].variable + [[0.2f]].variable
```

The output of the cell is:

```
Out[1]: Variable!(float, dim=2, HostStorage(T))(data=[0.3, 0.4, 0.5, 0.6], shape=[2, 2], strides=[2, 1])
```

Defining new autograd functions

Under the hood, each primitive autograd operator is really two functions that operate on Variables. The forward function computes output Variables from input Variable. The backward

⁵<https://github.com/ShigekiKarita/grain/blob/master/tutorial.ipynb>

grain is a lightweight library

smaller footprint library

framework	type	lines	mb
grain	static lib	12,431	0.6
chainer	python code	162,106	6
pytorch	dynamic lib	193,754	911
tensorflow	dynamic lib	130,475	285

smaller exe file (MNIST : 1.8MB, CIFAR: 2.3MB)

grain is fast

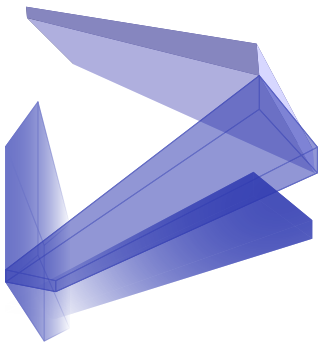
task	backend	framework	train (mb/s)	test (mb/s)
mnist	CUDA	grain	300	-
		chainer	-	-
		pytorch	200	-
	CPU	grain	90	-
		chainer	-	-
		pytorch	80	-
ptb	CUDA	grain	-	-
		pytorch	-	-

- ▶ grain 0.0.10, chainer 4.4.0, pytorch 0.4.1, CUDA9.2, CUDNN7

summary

grain: deep learning framework for D language

- ▶ **DYNAMIC**: like chainer and pytorch
- ▶ **SAFE**: statically typed variable and function
- ▶ **LIGHT**: simple like Python, small like C++
- ▶ **FAST**: mir and CUDA backend



CONTRIBUTION IS WELCOME!

<https://github.com/ShigekiKarita/grain>

examples

- ▶ Image recognition (mnist, cifar)
- ▶ Language modeling (shakespeare, ptb)
- ▶ WIP
 - ▶ Reinforcement learning (cartpole)
 - ▶ Speech recognition (librispeech)
 - ▶ Machine translation (anki)

future work

- ▶ probabilistic programming
- ▶ lazy evaluation mode
- ▶ low resource environment (RaspberryPi)